Proseminar Kryptographische Protokolle

DES und AES

Cyrill Kulka

Symmetrische Verfahren benutzen für die Verschlüsselung des Klartextes wie auch für die Entschlüsselung des Geheimtextes ein und denselben Geheimschlüssel. Darin liegt auch der große Nachteil, denn die Geheimhaltung der verschlüsselten Nachricht steht und fällt mit der sicheren und geheimen Übertragung des Schlüssels zwischen den beiden Kommunikationspartner.

1 Data Encryption Standard (DES)

Der Data Encryption Standard wurde ursprünglich von IBM entwickelt, doch bevor der sogenannte LUCIFER durch das NBS (National Bureau of Standards) zum neuen Verschlüsselungsstandards DES zertifiziert wurde, modifizierte die NSA den Algorithmus noch. Geändert wurde unter anderem die Schlüssellänge, indem sie von 128 Bit auf nur 56 Bit verkürzt wurde. 2000 wurde DES aufgrund Sicherheitsmängel von AES abgelöst. Bis heute wird DES allerdings, wenn auch in abgeänderter Form, immer noch verwendet.

1.1 Arbeitsweise des DES

DES verschlüsselt eine 64-Bit Eingabe E mit einem 64-Bit Schlüssel K zu einer 64-Bit Ausgabe.

Der Algorithmus ist eine rundenbasierende Feistelchiffre, d.h. der Eingabeblock wird in zwei Hälften zerlegt und getrennt voneinander in mehreren Runden verschlüsselt. In jeder Runde wird ein neuer Rundenschlüssel verwendet, der anfangs aus dem Schlüssel abgeleitet wird.

Bei der Berechnung der Teilschlüssel wird der 64-Bit Schlüssel auf nurmehr 56 Bit verkürzt, da die 8 Paritätbits aussortiert werden. Dieses Löschen verkleiner den Schlüsselraum jedoch erheblich von 2^{64} (~1,8 * 10^{19}) auf 2^{56} (~7,2 * 10^{16}) möglichen Schlüsselwerten.

Vor der Verschlüsselung wird der Eingabeblock einer Initialpermutation unterzogen. Anschließend folgen 16 Runden, die alle nach demselben Schema ablaufen. Als Vorbereitung wird der Eingabeblock E_0 in zwei 32-Bit Hälften zerlegt, L_0 und R_0.

Für die folgenden Hälften L_i und R_i mit i={1, ..., 16} gilt:

$$L_i = R_{i-1} \text{ und } R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i)$$

Die Funktion f ist das Herzstück von DES.

1. Der erste Schritt besteht aus einer Permutation, wobei zusätzlich einige Bits verdoppelt werden und sich die Größe von 32 Bit auf 48 Bit erweitert.
2. Das Ergebnis dieser sogenannten Expansion wird nun durch XOR mit dem dazugehörigen Rundenschlüssel K_i verknüpft.
3. Anschließend wird der Block in 6-Bit große Blöcke A_l mit l={1, ..., 8} aufgeteilt. A_l dient als Eingabe für die sogenannte S-Box S_l, welche den 6-Bit Block A_l durch einen 4 Bit großem Block B_l substituiert. B_l ist ein Eintrag in der 4x16 Matrix S_l. Das erste und letzte Bit von A_l bildet den Zeilenindex, die mittleren 4 Bits den Spaltenindex.
4. Alle Blöcke B_l werden konkateniert und, im 4. Schritt, nochmals permutiert. Die Permutation bildet das Ergebnis von f().

Die letzten beiden Schritte sind der Grund für die Stärke von DES. Durch die Permutation am Ende treten alle Bits untereinander in Wechselwirkung, d.h. aus einer minimalen Veränderung des Klartextes ergibt sich ein vollkommen anderer Geheimtext. Diese Eigenschaft wird auch Avalanche-Effekt genannt und erwirkt eine Diffusion, also das verschleiern von bestimmten Charakteristika, wie z.B. der Buchstabenhäufigkeit.

Nach der letzten, der 16. Runde werden die Hälften R16 und L16 nochmals vertauscht bevor sie

konkateniert werden. Nach der abschließenden inversen Initialpermutation wurde aus dem Eingabeblock ein Geheimtext erzeugt.

Die Entschlüsselung erfolgt nach demselben Prinzip der Verschlüsselung, mit einer Abweichung: Die Rundenschlüssel werden in umgekehrter Reihenfolge angewendet.

1.2 Weiterentwicklung von DES

Um die Schwäche des (zu) kleinen Schlüsselraumes zu eliminieren wurde nach einer Lösung für dieses Problem gesucht. Da DES, nicht wie z.B. Vektoren, keine abgeschlossenen Gruppen bildet gilt:

```
DES(DES(Klartext, Schlüssel 1), Schlüssel 2) != DES(Geheimtext, Schlüssel 3)
```

Daraus entwickelte sich später auch Triple-DES, auch DESede (DES encryption-decryption-encryption), welcher 1999 den Standard-DES ersetzte, bzw. als bevorzugte DES-Variante eingeführt wurde:

$$Geheimtext = DES(DES^{-1}(DES(Klartext, K_1), K_2), K_3)$$

Mit Schlüssel 1 K_1, Schlüssel 2 K_2 und Schlüssel 3 K_3

Die Verwendung von x Schlüsseln hat jedoch nicht zur Folge dass sich der Schlüsselraum um ein x-faches vergrößert. Durch sogenannte Meet-in-the-Middle Angriffe verringert sich die effektive Schlüssellänge bei Triple-DES auf nur 112 Bit. Im Gegensatz zum effektiven Schlüsselraum steigt der Rechenaufwand allerdings um ein x-faches.

2 Advanced Encryption Standard (AES)

Die wichtigsten Merkmale die vom NIST National Institute of Standards and Technology) gefordert und von AES auch erfüllt wurden sind foldende:

- leichte Implementierung in Hard- und Software
- Verwendung von 128-, 192- und 256-Bit Schlüssel
- 128-Bit Eingabeblöcke
- Widerstandsfähigkeit gegen bekannte Kryptoanalyseverfahren
- geringe Ressourcenanforderung, kurze Codelänge (für Einsatz in Smartcards)
- effizienter und sicherer als Triple-DES

2.1 Arbeitsweise des AES

Ebenso wie DES ist AES ein rundenbasierender Algorithmus. Die Anzahl der Runden r hängt von der Schlüssellänge ab, wobei auch hier für jede Runde ein eigener Schlüssel K_i verwendet wird. Diese Rundenschlüssel werden aus dem Schlüssel K rekursiv abgeleitet und sind nichtlinear und nichtinvertierbar.

Zuerst wird der Eingabeblock wird in ein zweidimensionales Array übertragen und mit dem Schlüssel K_0 mittels XOR verknüpft. Anschließend folgen r Runden, welche aus 4 Schritten bestehen:

1. Zuerst wird eine *monoalphabetische Substitution* durchgeführt. Diese dient in erster Linie zur Vermischung der Bytes um in Verbindung mit den folgenden Schritten einen möglichst großen Avalanche-Effekt zu erzielen. Hierbei kommt, wie bei DES, eine S-Box zum Einsatz.
2. Anschließend werden die Bytes in den Zeilen des Arrays im sogenannten *ShiftRow-Transformation* zyklisch nach links verschoben. Wie oft die Bytes einer Zeile in einer Runde verschoben werden ist abhängig von der Zeilennummer und der Blocklänge.
3. Im dritten Schritt, der *MixColumn-Transformation*, werden die Spalten vermischt. Jede Zelle einer Spalte wird neu berechnet, indem die Spalte einer Matrizenmultiplikation unterzogen wird, und das Ergebnis mittels XOR verknüpft wird. In diesem Schritt gehen alle Bits einer Spalte in eine Wechselbeziehung miteinander ein, das Ergebnis hängt von jedem einzelnen Bit ab. Der zugrundeliegende Zahlenraum ist ein Galois-Körper der Größe $GF(2^8)$.

   Dieser Schritt wird allerdings in der letzten Runde ausgelassen.
4. Abschließend wird am Ende in jeder Runde die *KeyAddition*, eine XOR-Verknüpfung des

Rundenschlüssels mit dem Block vorgenommen.

Durch diese Operationen, v.a. der MixColumn-Funktion, wird ein sehr starker Avalanche-Effekt erzielt, d.h. jedes Bit des Klartextes hängt von jedem Bit des Schlüssels ab. Nur eine kleine Veränderung des Schlüssels oder des Klartextes hat eine starke Veränderung des Geheimtextes zur Folge.

Das Ergebnis der r-ten Runde ist der Geheimtext.

2.2 Vorgehensweise bei der Entschlüsselung

Die Entschlüsselung erfolgt durch den invertierten Ablauf der Verschlüsselung. Hierbei werden allerdings nicht nur die Abfolge der Arbeitsschritte, sondern zusätzlich auch die Richtung der Shift-Row-, der MixColumn-Transformation und der Substitution umgekehrt.

3 Anwendungen der Verschlüsselungsmethoden

Da die vorgestellten Verschlüsselungsmethoden nur Eingabeblöcke fester Länge erwarten, bzw. nur diese verarbeiten können, müssen sie in eine Umgebung eingebettet werden, Verschlüsselungsalgorithmen mit artgerechten Häppchen „füttert“. Diese Umgebungen können anhand ihrer Arbeitsweise in fünf verschiedene Typen aufgeteilt werden

3.1 Blockchiffre

Die Blockchiffre kann nur verwendet werden, wenn die Größe des zu verschlüsselnden Klartextes bekannt ist. Die Vorgehensweise ist im Grunde dieselbe: ein Klartext wird Eingabeblöcke der richtigen Länge zerlegt und der Letzte gegebenenfalls mittels Padding aufgefüllt.

3.1.1 Electronic Code Book: ECB

Der ECB-Modus verschlüsselt alle Klartextblöcke unabhängig voneinander. Dieser Modus ist der unsicherste von allen Modi, Grund dafür sind die immer gleichen Geheimtextmuster welche gleiche Klartextblöcke generieren. Ein Angreifer kann die Geheimtextblöcke unabhängig voneinander analysieren und mithilfe von vorhandenen oder vermuteten Klartext/Geheimtextpaaren Rückschlüsse auf den Schlüssel ziehen. Die Entschlüsselung kann in einer beliebigen Reihenfolge vonstatten gehen.

3.1.2 Cipher Block Chaining: CBC

Diese Schwäche wird im CBC-Modus eliminiert, indem die Klartextblöcke vor der Verschlüsselung mit dem vorherigen Geheimtextblock mittels XOR verknüpft werden. Für einen Klartextblock M_i aus dem Klartext M mit einer Verschlüsselung durch einen Verschlüsselungsalgorithmus D() gilt also:

$$C_i = D(\text{Klartext } M_i \text{ XOR Geheimtext } C_{i-1}, \text{ Schlüssel K})$$

Für den ersten Klartextblock wird zur XOR-Verknüpfung ein vereinbarter sogenannter Initialisierungsvektor benutzt. Die Verkettung bewirkt allerdings auch, dass Übertragungsfehler zu einer falschen Entschlüsselung des betroffenen und des Nachfolgenden Geheimtextblocks auswirkt. Auch muss die Entschlüsselung der Geheimtextblöcke in derselben Reihenfolge vorgenommen werden wie bei der Verschlüsselung.

3.2 Stromchiffre

Stromchiffren verschlüsseln Klartextströme (z.B. bei Telefonaten oder Chats), welche in der Regel bitweise kodiert werden. Auch hier ist die Vorgehensweise innerhalb der einzelnen Modi ähnlich.

Um den Klartextstrom zu verschlüsseln wird er mit einem genauso langen Schlüsselstrom mittels XOR verknüpft. Die Sicherheit beruht auf einem Pseudozufallsgenerator, der eine Schlüsselreihe mit der Eigenschaft einer echt zufälligen Folge produziert. Für diesen Pseudozufallsgenerator werden Blockchiffren, also z.B. DES oder AES, eingesetzt. Beide Kommunikationspartner kennen den Schlüssel K und einen Initialisierungsvektor IV, welche zur Erzeugung des Schlüsselstroms nötig sind und benutzen denselben Pseudozufallsgenerator. Der Schlüsselstrom für die erste Runde ist der kodierte Initialisierungsvektor, für die folgenden Runden wird der Initialisierungsvektor nach einem bestimmten Schema modifiziert um so neue Eingabeblöcke für die Blockchiffre zu erzeugen neue Schlüsselströme zu generieren.

3.2.1 Output Feedback (OFB)

Der Initialisierungsvektor wird aus dem Datum gebildet. Der Eingabeblock wird dahingehend modifiziert als dass die Bits, die zur Verschlüsselung der Klartextstroms verwendet wurden, per Schieberegister in den Eingabeblock geschoben werden. Bei Synchronisationsfehlern kann der Empfänger aufgrund des falschen Datums für den Initialisierungsvektor keine Geheimtextblöcke dechiffrieren. Übertragungsfehler wirken sich jedoch nur auf die betroffene Stelle aus.

Die Dekodierung erfolgt nach genau demselben Schema wie die Verschlüsselung, nur dass Geheimtextstrom und Klartextstrom die Plätze tauschen. Auch hierbei muss aufgrund der Verkettung, wie schon bei dem CBC-Modus, auf die Reihenfolge geachtet werden.

3.2.2 Cipher Feedback (CFB)

Der Unterschied zum OFB ist nur marginal, nur die Beschaffung der Bits zur Veränderung des Initialisierungs-, bzw. Eingabeblocks ist anders: Anstatt der Schlüsselbits werden die Geheimtextbits verwendet. Durch diese Verkettung wirken sich Übertragungsfehler noch schwerwiegender aus.

3.2.3 Counter-Modus

Im Gegensatz zum CFB- oder OFB-Modus verwendet der Counter-Modus keine Bits aus dem Klartext- oder Geheimtextstrom, sondern benutzt einen Zahlenwert der zu Beginn zwischen den beiden Kommunikationspartnern vereinbart wurde und erhöht diesen nach jeder Runde um Eins. Hierbei muss einzig und allein beachtet werden dass sich die Zahlen nicht wiederholen. Ein Vorteil dieser Methode ist, dass man die Geheimtextblöcke unabhängig voneinander entschlüsseln kann, einzig und allein der Indexwert des Geheimtextblockes wird benötigt.

4 Literatur

- http://www.codeplanet.eu/tutorials/cpp/3-cpp/51-advanced-encryption-standard.html
- http://www-rn.informatik.uni-bremen.de/lehre/itsec/itsec05-4a.pdf
- Claudia Eckert: *IT-Sicherheit: Konzepte, Verfahren, Protokolle* 5. überarbeitete Auflage, R. Oldenbourg-Verlag, 2007